(19) European Patent Office
Office européen des brevets

(11) EP 0 848 314 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.06.1998 Bulletin 1998/25

(51) Int. Cl.^{6}: G06F 1/00, G06F 12/14

(21) Application number: 97306103.9

(22) Date of filing: 11.08.1997

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: 11.12.1996 US 763917

(71) Applicant:
NCR INTERNATIONAL INC.
Dayton, Ohio 45479 (US)

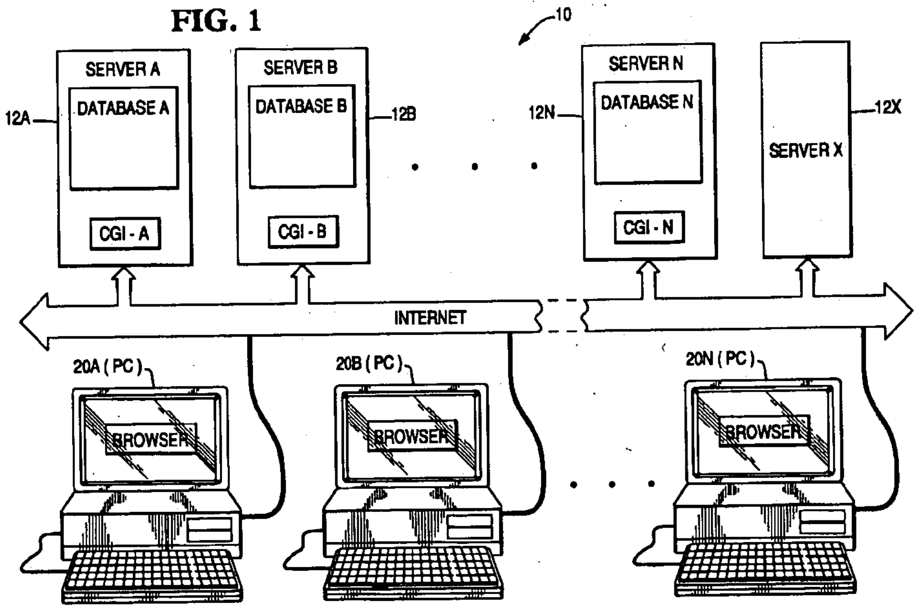
(72) Inventors:
- McCollum, Tab
Camden, Ohio 45311 (US)
- Jury, Thomas W.
Beavercreek, Ohio 45440 (US)

(74) Representative:
Irish, Vivien Elizabeth
International IP Department,
NCR Limited,
206 Marylebone Road
London NW1 6LY (GB)

(54) Document security system and method

(57) A server (A) has a database (12A) in which a number of documents are stored. The server (A) is accessible by many distant user terminals (20A..20N), for example via the internet. The documents may be in HTML form for distribution through the World Wide Web. Each document is assigned a security level and different documents may have different security levels on a document by document basis. The database (12A) includes a table containing a user name, a password and a security level indicator. When a particular document is requested by a user terminal (20A) the security level of the user is determined from the table and access to the document is allowed only if the security level of the requesting user is as least as high as the security level of the requested document.

FIG. 1



Description

This invention relates to a document security system and method. It has application in the provision of a security database for HTML (hypertext markup language) documents in a World Wide Web application.

With the increased number of internet users and the ease of accessibility of the World Wide Web, there is an increasing demand for the use of the Web as a vehicle for distributed applications. These distributed applications are composed of HTML documents and can be accessed by various Web browsers, such as Netscape Navigator or Microsoft Internet Explorer. Hypertext links relate the documents to each other and give users a way to navigate from one file to another.

These distributed applications require security to limit access to valid users. Currently, a typical approach to providing security for HTML documents requires the server directory and subdirectories where the HTML documents are located to be secured at the same level. This means that an individual user can have access to all the documents in the directory or access to none of the documents in the directory based on an appropriate user id and password. Another drawback of this typical approach is that this approach depends upon the naming convention used for the subdirectories and thus makes porting of the application (and all of the associated HTML documents) to another server difficult.

It is an object of the invention to provide security of access to individual documents on a document-by-document basis.

According to the invention in one aspect a document security system comprising a server in which a plurality of documents are stored for access by user terminals is characterised in that a database is provided in the server, which database has: means for storing user information; means for storing document information; and means for providing access to the stored documents document-by-document on the basis of the user information and the document information.

The said means for storing user information may include means for storing a user identification name, an associated user password and an associated security level indicator for indicating the highest level of security access for the user name associated therewith.

The said means for storing document information may include a file name, code means for creating a document associated with the file name and a security level indicator associated with the file name for indicating the security level of the associated document.

The said means for providing access to stored documents may be included in a common gateway interface file.

In carrying out the invention a plurality of different servers may be provided each having its own database and each having an internet connection to enable any of a plurality of user terminals to be connected to any of the servers.

According to the invention in another aspect method of providing document security in an environment where a server stores a plurality of documents and the server is accessible by any of a plurality of user terminals comprising the steps of: assigning a security level to each document, assigning a security level to each user terminal, receiving a request at the server from a user terminal for access to a document, determining the security level assigned to the user terminal, comparing the determined security level with the security level assigned to the requested document, and providing access to the requested document only if the result of the comparison step indicates that the security level of the said user terminal is at least as high as the security level assigned to the requested document.

A plurality of servers may be provided in which case there may be included the step of locating the particular server in which the requested document is stored.

In embodiments of the invention there may be included the step of associating a user identification name and a user password with the assigned user security level.

The invention is readily applicable to providing security for HTML documents in a world wide web application. Such security is available to control user access to individual HTML documents or groups of documents. Furthermore the applications, or documents in an application, can be readily ported to other servers since the applications do not rely on directory structure to provide security.

The invention will now be described by way of example with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram of a system of the present invention;
Fig. 2 is a block diagram of a User Table for use with the present invention;
Figs. 3A and 3B are block diagrams of a File Table for use with the present invention; and
Fig. 4 is a flowchart of the method of the present invention.

A portion of the disclosure of this patent document contains material which is subject to copyright protection and to which a claim of copyright protection is made. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyrights and similar rights whatsoever.

Referring now to the drawings, in which like-referenced characters indicate corresponding elements throughout the several views, attention is first drawn to Figure 1 which shows a block diagram of the system 10 for providing a security database for HTML documents in a WWW application. The system 10 includes a plurality of PCs 20A through 20N or other client terminals which have access to the Internet. PCs 20A through 20N include a web browser such as Netscape Navigator or Microsoft Internet Explorer. PCs 20A through 20N also include an input device such as a keyboard or a mouse and other standard components such as memory, display, microprocessor, etc.

The system 10 also includes a plurality of servers 12A through 12X or other large storage devices also connected to the Internet. Each server 12A, 12B , ... 12N may include a database having specified files or the server may not initially include a database at all such as 12X. The databases included may be any commonly available databases. Examples include Access (available from Microsoft), Dbase (available for Ashton-Tate), etc. Each server 12A, 12B, ... 12N also includes a CGI (common gateway interface) script file (CGI-A, CGI-B, etc.) for passing information from the PCs via the browsers to the servers and from the servers via the browsers to the PCs. The required CGI script files can be built with just about any programming or scripting language (for example, C) that the user's servers support. The CGI code provides the interface with the server and passes and receives the information between the database in the server and the user terminal. A sample of CGI code is included at the end of this description. The sample CGI code also includes an invention-specific or customized module entitled "Module 2" which provides specific examples of code for checking security levels and granting access and downloading HTML files according to the present invention.

In discussing Figs. 2, 3A and 3B, exemplary database A in server 12A will be described. This description applies to the other databases located in the other servers. Additionally, server 12X does not initially include a database. However, according to the present invention, any desired database from one of the other servers can be ported to server 12X without the difficulties normally encountered when moving or copying a grouping of files from one server to another server when the files are located in the directory structure.

Referring first to Fig. 2, database A includes a User Table 100 which is basically used to keep track of users. The User Table may include fields such as user name 110, user identification (id) 120, user password 130, user security level 140 and user group 150. Additional fields may be included or some of the above fields may be deleted as long as the User Table contains enough information to accurately identify a user requesting a document and provide the security level (or privileges) corresponding to that user.

Referring next to Figs. 3A and 3B, database A also includes a Files Table 200 which is basically used to control access to individual HTML documents. The Files Table 200 may include fields such as security level 210, user group 220, file name 230 and HTML code 240. Additional fields may be included or some of the above fields may be deleted as long as the Files Table contains enough information to accurately determine if a requested document is contained in the database and whether a user requesting a document should be given access to the document. If the user is given access, then the code (or file) in the HTML code field 240 is passed to the user (through the user's browser). To provide the HTML code field to the user, the customized CGI module passes the code (or file) to the user verbatim with the following exception. In any hypertext links to other documents, referenced by ⟨A HREF=filename⟩ HTML tags, the specific file name is replaced with a reference to the customized CGI module and the file name is appended as a parameter.

For example:

⟨A HREF="filename" is replaced with
⟨A HREF="invent.exe?Name=UserName&file=filename"

In this way the customized CGI module can interact with the user's web browser and invoke the correct hypertext link to other files in the database. This process allows the passing of the file to the user to occur without any noticeable difference from a server with security protection for the entire server or subdirectory because the user's inquiry for a specific HTML file calls the customized CGI module which handles the processing of the user's security level and user group and the file's required security level and user group. Thus, although the present invention provides flexibility in allowing access to various documents on a server, the user interface is virtually the same as standard systems which do not provide varying security levels for documents on the same server or in the same directory.

Fig. 4 shows a flowchart of the method for providing a security database for HTML documents in WWW applications. First in step 310, a user requests access to a file, preferably using a web browser. Then in step 320, the web browser interacts with cgi script files in servers 12A through 12N until the desired file, which is embedded in a database, is located in a particular server. Alternatively, a user could request a list of all files located in a particular database and select a desired file from that list. Next in step 330, the cgi script file of the particular server uses the user id and the user password to determine the assigned security level and user group in the User Table 100.

Then in step 340, the cgi script file compares the user's security level and user group with those required in the Files Table 200 corresponding to the desired file. In step 350, it is determined whether the user has the required security level and user group to access the desired file. If yes, then in step 360, the information in the HTML Code field 240 of Files Table 200 is provided to the user's browser as described above. Thus the user is provided a first document or web page. If the user requests a different file in the same server then in step 370, the step of comparing the user's security level and user group with those required in the Files Table 200 corresponding to the new desired file is performed. If the user does not request a different file in the same server then the process is ended in step 390. (Of course, If the user requests access to a file in another server, then the user's browser must interact with the cgi script files in all the servers until the desired file embedded in the database of the new server is located.)

If a user requests access to a document and does not have the required security level and user group for the desired document, then the user is informed that access has been denied in step 380. Then the process ends in step 390.

An advantage of the present invention is that user access to individual HTML documents (or groups of documents) can be determined and controlled.

Another advantage of the present invention is that applications (or documents in an application) can be ported to other servers since the applications do not rely on the directory structure to provide security. Rather the documents are located in the database.

Although the invention has been described with the use of an example CGI script file and related customized module of the CGI file, it is contemplated that any coding which provides the functions as discussed with respect to the above files is contemplated within the scope of the present invention. Additionally, although the program providing the fields has been described as a database, any program which can provide fields to be accessed and compared according to the description is contemplated within the scope of the present invention.

```
'
'Program Name:  invent.Exe
'Date: November 1996
'
'Author:   Tab McCollum
'          NCR Corporation
'          Information Products
'          Research and Development
'
'Programming Language:  Visual Basic 4
'Program Sources Files:
'                       cgi32.bas
'                       dbsample.bas
'                       invent.vbp
'other files needed:
'                       db1.mdb
'
'Program Purpose:  This program can only be used in conjunction with
'a world wide web server that supports the windows cgi specifcation.
'
'This program provides a secure means of taking html files that have
'been stored in the db1.mdb database file in the files table and
'restricting access to them.
'
'The program first lists an index of the files available and allows the
'user to select a file name.  At that time the user also inputs a
'user name and password which is then sent to the www server.
'
'The program then validates the user by password, security level and
'group level before the html file is displayed.
'
'security and group levels are required for both users and files
'
'
'Please note that this is all done within the database itself and does
'not rely on the security mechanisms of the web server.
'
'
'Notice:  The author is not responsible for the data content that is the
'result of using this program.
'
'
'The source code in the CGI32.Bas file below is a freely distributed file.
'It is not covered by any copyright notices.
'
'****Copyrights*****
'All source code in the dbsample.bas file is copyrighted as described below:
'
'Copyright NCR Corporation, all rights reserved 1996
'
'-------------------------------------------------------------------
'       *************
'       * CGI32.BAS *
'       *************
'
' VERSION: 1.7  (December 3, 1995)
'
' AUTHOR:  Robert B. Denny <rdenny@netcom.com>
'
' Common routines needed to establish a VB environment for
' Windows CGI programs that run behind the WebSite Server.
'
' INTRODUCTION
'
' The Common Gateway Interface (CGI) version 1.1 specifies a minimal
' set of data that is made available to the back-end application by
' an HTTP (Web) server. It also specifies the details for passing this
' specific to Unix-like environments. The NCSA httpd for Windows does
' supply the data items (and more) specified by CGI/1.1, however it
' uses a different method for passing the data to the back-end.
'
' DEVELOPMENT
'
' WebSite requires any Windows back-end program to be an
' executable image. This means that you must convert your VB
' application into an executable (.EXE) before it can be tested
' with the server.
'
' ENVIRONMENT
'
' The WebSite server executes script requests by doing a
' CreateProcess with a command line in the following form:
'
'   prog-name cgi-profile
'
' THE CGI PROFILE FILE
'
' The Unix CGI passes data to the back end by defining environment
' variables which can be used by shell scripts. The WebSite
' server passes data to its back end via the profile file. The
' format of the profile is that of a Windows ".INI" file. The keyword
' names have been changed cosmetically.
'
' There are 7 sections in a CGI profile file, [CGI], [Accept],
' [System], [Extra Headers], and [Form Literal], [Form External],
' and [Form huge]. They are described below:
'
' [CGI]                    <== The standard CGI variables
' CGI Version=             The version of CGI spoken by the server
' Request Protocol=        The server's info protocol (e.g. HTTP/1.0)
' Request Method=          The method specified in the request (e.g., "GET")
' Request Keep-Alive=      If the client requested connection re-use (Yes/No)
' Executable Path=         Physical pathname of the back-end (this program)
' Logical Path=            Extra path info in logical space
' Physical Path=           Extra path info in local physical space
' Query String=            String following the "?" in the request URL
' Content Type=            MIME content type of info supplied with request
' Content Length=          Length, bytes, of info supplied with request
' Request Range=           Byte-range specfication received with request
' Server Software=         Version/revision of the info (HTTP) server
' Server Name=             Server's network hostname (or alias from config)
' Server Port=             Server's network port number
' Server Admin=            E-Mail address of server's admin. (config)
' Referer=                 URL of referring document
' From=                    E-Mail of client user (rarely seen)
' User Agent=              String describing client/browser software/version
' Remote Host=             Remote client's network hostname
' Remote Address=          Remote client's network address
' Authenticated Username=Username if present in request
' Authenticated Password=Password if present in request
' Authentication Method=Method used for authentication (e.g., "Basic")
' Authentication Realm=Name of realm for users/groups
'
' [Accept]                 <== What the client says it can take
' The MIME types found in the request header as
'    Accept: xxx/yyy; zzzz...
' are entered in this section as
'    xxx/yyy=zzzz...
' If only the MIME type appears, the form is
'    xxx/yyy=Yes
'
' [System]                 <== Windows interface specifics
' GMT Offset=              Offset of local timezone from GMT, seconds (LONG!)
' Output File=             Pathname of file to receive results
' Content File=            Pathname of file containing raw request content
' Debug Mode=          If server's CGI debug flag is set (Yes/No)
'
' [Extra Headers]
' Any "extra" headers found in the request that activated this
' program. They are listed in "key=value" form. Usually, you'll see
' at least the name of the browser here as "User-agent".
'
' [Form Literal]
' If the request was a POST from a Mosaic form (with content type of
' "application/x-www-form-urlencoded"), the server will decode the
' form data. Raw form input is of the form "key=value&key=value&...",
' with the value parts "URL-encoded". The server splits the key=value
' pairs at the '&', then spilts the key and value at the '=',
' URL-decodes the value string and puts the result into key=value
' (decoded) form in the [Form Literal] section of the INI.
'
' [Form External]
' If the decoded value string is more than 254 characters long,
' or if the decoded value string contains any control characters
' or quote marks the server puts the decoded value into an external
' tempfile and lists the field in this section as:
'    key=<pathname> <length>
' where <pathname> is the path and name of the tempfile containing
' the decoded value string, and <length> is the length in bytes
' of the decoded value string.
'
' NOTE: BE SURE TO OPEN THIS FILE IN BINARY MODE UNLESS YOU ARE
'       CERTAIN THAT THE FORM DATA IS TEXT!
'
' [Form File]
' If the form data contained any uploaded files, they are described in
' this section as:
'    key=[<pathname>] <length> <type> <encoding> [<name>]
' where <pathname> is the path and name of the tempfile contining the
' uploaded file, <length> is the length in bytes of the uploaded file,
' <type> is the content type of the uploaded file as sent by the browser
' <encoding> is the content-transfer encoding of the uploaded file, and
' <name> is the original file name of the uploaded file.
'
' [Form Huge]
' If the raw value string is more than 65,536 bytes long, the server
' does no decoding. In this case, the server lists the field in this
' section as:
'    key=<offset> <length>
' where <offset> is the offset from the beginning of the Content File
' at which the raw value string for this key is located, and <length>
' is the length in bytes of the raw value string. You can use the
' <offset> to perform a "Seek" to the start of the raw value string,
' and use the length to know when you have read the entire raw string
' into your decoder. Note that VB has a limit of 64K for strings, so
'
' Examples:
'
'    [Form Literal]
'    smallfield=123 Main St. #122
'
'    [Form External]
'    field300chars=c:\website\cgi-tmp\1a7fws.000 300
'    fieldwithlinebreaks=c:\website\cgi-tmp\1a7fws.001 43
'
'    [Form Huge]
'    field230K=c:\website\cgi-tmp\1a7fws.002 276920
'
' =====
' USAGE
' =====
' Include CGI32.BAS in your VB4 project. Set the project options for
' "Sub Main" startup. The Main() procedure is in this module, and it
' handles all of the setup of the VB CGI environment, as described
' above. Once all of this is done, the Main() calls YOUR main procedure
' which must be called CGI Main(). The output file is open, use Send()
' to write to it. The input file is NOT open, and "huge" form fields
' have not been decoded.
'
' NOTE: If your program is started without command-line args,
' the code assumes you want to run it interactively. This is useful
' for providing a setup screen, etc. Instead of calling CGI Main(),
' it calls Inter Main(). Your module must also implement this
' function. If you don't need an interactive mode, just create
' Inter Main() and put a 1-line call to MsgBox alerting the
' user that the program is not meant to be run interactively.
' The samples furnished with the server do this.
'
' If a Visual Basic runtime error occurs, it will be trapped and result
' in an HTTP error response being sent to the client. Check out the
' Error Handler() sub. When your program finishes, be sure to RETURN
' TO MAIN(). Don't just do an "End".
'
' Have a look at the stuff below to see what's what.
'
'----------------------------------------------------------------------
' Author:   Robert B. Denny <rdenny@netcom.com>
'           April 15, 1995
'
' Revision History:
'   15-Apr-95 rbd   Initial release (ref VB3 CGI.BAS 1.7)
'   02-Aug-95 rbd   Changed to take input and output files from profile
'                   Server no longer produces long command line.
'   24-Aug-95 rbd   Make call to GetPrivateProfileString conditional
'                   so 16-bit and 32-bit versions supported. Fix
'                   computation of CGI GMTOffset for offset=0 (GMT)
'                   case. Add FieldPresent() routine for checkbox
'                   handling. Clean up comments.
'   29-Oct-95 rbd   Added PlusToSpace() and Unescape() functions for
'                   decoding query strings, etc.
'   16-Nov-95 rbd   Add keep-alive variable, file uploading description
'                   in comments, and upload display.
'   20-Nov-95 rbd   Fencepost error in ParseFileValue()
'   23-Nov-95 rbd   Remove On Error Resume Next from error handler
'   03-Dec-95 rbd   User-Agent is now a variable, real HTTP header
'                   Add Request-Range as http header as well.
'----------------------------------------------------------------------
Option Explicit
'
' ==================
' Manifest Constants
' ==================
'
Const MAX CMDARGS = 8          ' Max # of command line args
Const ENUM BUF SIZE = 4096     ' Key enumeration buffer, see GetProfile()
' These are the limits in the server
Const MAX XHDR = 100           ' Max # of "extra" request headers
Const MAX ACCTYPE = 100        ' Max # of Accept: types in request
Const MAX FORM TUPLES = 100    ' Max # form key=value pairs
Const MAX HUGE TUPLES = 16     ' Max # "huge" form fields
Const MAX_FILE_TUPLES = 16     ' Max # of uploaded file tuples
'
'
' =====
' Types
' =====
'
Type Tuple                     ' Used for Accept: and "extra" headers
    key As String              ' and for holding POST form key=value pairs
    value As String
```

```
'
                                      ' Used for form-based file uploads
Type FileTuple                        ' Form field name
    key As String                     ' Local tempfile containing uploaded file
    file As String                    ' Length in bytes of uploaded file
    length As Long                    ' Content type of uploaded file
    type As String                    ' Content-transfer encoding of uploaded file
    encoding As String                ' Original name of uploaded file
    name As String
End Type
                                      ' Used for "huge" form fields
Type HugeTuple                        ' Keyword (decoded)
    key As String                     ' Byte offset into Content File of value
    offset As Long                    ' Length of value, bytes
    length As Long
End Type
'
'
' ================
' Global Constants
' ================
'
' -----------
' Error Codes
' -----------
'
Global Const ERR ARGCOUNT = 32767              ' HTTP 400
Global Const ERR BAD REQUEST = 32766           ' HTTP 401
Global Const ERR UNAUTHORIZED = 32765          ' HTTP 402
Global Const ERR PAYMENT REQUIRED = 32764      ' HTTP 403
Global Const ERR FORBIDDEN = 32763             ' HTTP 404
Global Const ERR NOT FOUND = 32762             ' HTTP 500
Global Const ERR INTERNAL ERROR = 32761        ' HTTP 501
Global Const ERR NOT IMPLEMENTED = 32760       ' HTTP 503 (experimental)
Global Const ERR TOO BUSY = 32758              ' GetxxxField "no field"
Global Const ERR NO FIELD = 32757              ' Start of our errors
Global Const CGI_ERR_START = 32757

' ===================
' CGI Global Variables
' ===================
'
' ---------------------
' Standard CGI variables
' ---------------------
'
Global CGI ServerSoftware As String
Global CGI ServerName As String
Global CGI ServerPort As Integer
Global CGI RequestProtocol As String
Global CGI ServerAdmin As String
Global CGI Version As String
Global CGI RequestMethod As String
Global CGI RequestKeepAlive As Integer
Global CGI LogicalPath As String
Global CGI PhysicalPath As String
Global CGI ExecutablePath As String
Global CGI QueryString As String
Global CGI RequestRange As String
Global CGI Referer As String
Global CGI From As String
Global CGI UserAgent As String
Global CGI RemoteHost As String
Global CGI RemoteAddr As String
Global CGI AuthUser As String
Global CGI AuthPass As String
Global CGI AuthType As String
Global CGI AuthRealm As String
Global CGI ContentType As String
Global CGI_ContentLength As Long
'
' ------------------
' HTTP Header Arrays
' ------------------
'
Global CGI AcceptTypes(MAX ACCTYPE) As Tuple       ' Accept: types
Global CGI NumAcceptTypes As Integer               ' # of live entries in array
Global CGI ExtraHeaders(MAX XHDR) As Tuple         ' "Extra" headers
Global CGI_NumExtraHeaders As Integer              ' # of live entries in array
'
' --------------
' POST Form Data
' --------------
'
Global CGI FormTuples(MAX FORM TUPLES) As Tuple    ' POST form key=value pairs
Global CGI NumFormTuples As Integer                ' # of live entries in array
Global CGI HugeTuples(MAX HUGE TUPLES) As HugeTuple ' Form "huge tuples
Global CGI NumHugeTuples As Integer                ' # of live entries in array
Global CGI FileTuples(MAX FILE TUPLES) As FileTuple ' File upload tuples
Global CGI_NumFileTuples As Integer                ' # of live entries in array
'
' ----------------
' System Variables
' ----------------
'
Global CGI GMTOffset As Variant          ' GMT offset (time serial)
Global CGI ContentFile As String         ' Content/Input file pathname
Global CGI OutputFile As String          ' Output file pathname
Global CGI_DebugMode As Integer          ' Script Tracing flag from server
'
'
' ========================
' Windows API Declarations
' ========================
'
' NOTE: Declaration of GetPrivateProfileString is specially done to
' permit enumeration of keys by passing NULL key value. See GetProfile().
' Both the 16-bit and 32-bit flavors are given below. We DO NOT
' recommend using 16-bit VB4 with WebSite!
'
#If Win32 Then
Declare Function GetPrivateProfileString Lib "kernel32" _
    Alias "GetPrivateProfileStringA"
   (ByVal lpApplicationName As String, _
    ByVal lpKeyName As Any,
    ByVal lpDefault As String,
    ByVal lpReturnedString As String, _
    ByVal nSize As Long,
    ByVal lpFileName As String) As Long
#Else
Declare Function GetPrivateProfileString Lib "Kernel" _
   (ByVal lpSection As String, _
    ByVal lpKeyName As Any,
    ByVal lpDefault As String,
    ByVal lpReturnedString As String, _
    ByVal nSize As Integer,
    ByVal lpFileName As String) As Integer
#End If
'
'
' ===============
' Local Variables
' ===============
'
Dim CGI ProfileFile As String            ' Profile file pathname
Dim CGI OutputFN As Integer              ' Output file number
Dim ErrorString As String
```

```
'-----------------------------------------------------------------------
'
' Return True/False depending on whether a form field is present.
' Typically used to detect if a checkbox in a form is checked or
' not. Unchecked checkboxes are omitted from the form content.
'
'-----------------------------------------------------------------------
Function FieldPresent(key As String) As Integer
    Dim i As Integer

    FieldPresent = False                ' Assume failure

    For i = 0 To (CGI NumFormTuples - 1)
        If CGI FormTuples(i).key = key Then
            FieldPresent = True         ' Found it
            Exit Function               ' ** DONE **
        End If
    Next i                              ' Exit with FieldPresent still False

End Function


'-----------------------------------------------------------------------
'
'   ErrorHandler() - Global error handler
'
' If a VB runtime error occurs dusing execution of the program, this
' procedure generates an HTTP/1.0 HTML-formatted error message into
' the output file, then exits the program.
'
' This should be armed immediately on entry to the program's main()
' procedure. Any errors that occur in the program are caught, and
' an HTTP/1.0 error messsage is generated into the output file. The
' presence of the HTTP/1.0 on the first line of the output file causes
' NCSA httpd for WIndows to send the output file to the client with no
' interpretation or other header parsing.
'-----------------------------------------------------------------------
Sub ErrorHandler(code As Integer)

    Seek #CGI OutputFN, 1      ' Rewind output file just in case
    Send ("HTTP/1.0 500 Internal Error")
    Send ("Server: " + CGI ServerSoftware)
    Send ("Date: " + WebDate(Now))
    Send ("Content-type: text/html")
    Send ("")
    Send ("<HTML><HEAD>")
    Send ("<TITLE>Error in " + CGI_ExecutablePath + "</TITLE>")
    Send ("</HEAD><BODY>")
    Send ("<H1>Error in " + CGI ExecutablePath + "</H1>")
    Send ("An internal Visual Basic error has occurred in " + CGI_ExecutablePatt
 + ".")
    Send ("<PRE>" + ErrorString + "</PRE>")
    Send ("<I>Please</I> note what you were doing when this problem occurred,")
    Send ("so we can identify and correct it. Write down the Web page you were u
sing,")
    Send ("any data you may have entered into a form or search box, and")
    Send ("anything else that may help us duplicate the problem. Then contact th
e")
    Send ("administrator of this service: ")
    Send ("<A HREF=""mailto:" & CGI ServerAdmin & """>")
    Send ("<ADDRESS>&lt;" + CGI_ServerAdmin + "&gt;</ADDRESS>")
    Send ("</A></BODY></HTML>")

    Close #CGI_OutputFN
    End                  ' Terminate/the program
    '======
End Sub
'-----------------------------------------------------------------------
'
'   GetAcceptTypes() - Create the array of accept type structs
'
' Enumerate the keys in the [Accept] section of the profile file,
' then get the value for each of the keys.
'-----------------------------------------------------------------------
Private Sub GetAcceptTypes()
    Dim sList As String
    Dim i As Integer, j As Integer, l As Integer, n As Integer

    sList = GetProfile("Accept", "") ' Get key list
                                              ' Length incl. trailing null
    l = Len(sList)                            ' Start at 1st character
    i = 1                                     ' Index in array
    n = 0                                     ' Safety stop here
    Do While ((i < l) And (n < MAX ACCTYPE))  ' J -> next null
        j = InStr(i, sList, Chr$(0))
        CGI AcceptTypes(n).key = Mid$(sList, i, j - i) ' Get Key, then value
        CGI AcceptTypes(n).value = GetProfile("Accept", CGI_AcceptTypes(n).key)
                                              ' Bump pointer
        i = j + 1                             ' Bump array index
        n = n + 1
    Loop                                      ' Fill in global count
    CGI_NumAcceptTypes = n

End Sub
'-----------------------------------------------------------------------
'
'   GetArgs() - Parse the command line
'
' Chop up the command line, fill in the argument vector, return the
' argument count (similar to the Unix/C argc/argv handling)
'-----------------------------------------------------------------------
Private Function GetArgs(argv() As String) As Integer
    Dim buf As String
    Dim i As Integer, j As Integer, l As Integer, n As Integer

    buf = Trim$(Command$)                     ' Get command line

                                              ' Length of command line
    l = Len(buf)                              ' If empty
    If l = 0 Then                             ' Return argc = 0
        GetArgs = 0
        Exit Function
    End If

                                              ' Start at 1st character
    i = 1                                     ' Index in argvec
    n = 0                                     ' Safety stop here
    Do While ((i < l) And (n < MAX CMDARGS))  ' J -> next space
        j = InStr(i, buf, " ")                ' Exit loop on last arg
        If j = 0 Then Exit Do                 ' Get this token, trim it
        argv(n) = Trim$(Mid$(buf, i, j - i))  ' Skip that blank
        i = j + 1                             ' Skip any additional whitespace
        Do While Mid$(buf, i, 1) = " "
            i = i + 1
        Loop                                  ' Bump array index
        n = n + 1
    Loop

    argv(n) = Trim$(Mid$(buf, i, (l - i + 1))) ' Get last arg
    GetArgs = n + 1                           ' Return arg count

End Function
'-----------------------------------------------------------------------
```

```
'
'   GetExtraHeaders() - Create the array of extra header structs
'
' Enumerate the keys in the [Extra Headers] section of the profile file,
' then get the value for each of the keys.
'-----------------------------------------------------------------------
Private Sub GetExtraHeaders()
    Dim sList As String
    Dim i As Integer, j As Integer, l As Integer, n As Integer

    sList = GetProfile("Extra Headers", "")  ' Get key list
    l = Len(sList)                            ' Length incl. trailing null
    i = 1                                     ' Start at 1st character
    n = 0                                     ' Index in array
    Do While ((i < l) And (n < MAX XHDR))     ' Safety stop here
        j = InStr(i, sList, Chr$(0))          ' J -> next null
        CGI ExtraHeaders(n).key = Mid$(sList, i, j - i) ' Get Key, then value
        CGI_ExtraHeaders(n).value = GetProfile("Extra Headers", CGI_ExtraHeader:
(n).key)                                      ' Bump pointer
        i = j + 1                             ' Bump array index
        n = n + 1
    Loop
    CGI_NumExtraHeaders = n                   ' Fill in global count

End Sub
'-----------------------------------------------------------------------
'
'   GetFormTuples() - Create the array of POST form input key=value pairs
'
'-----------------------------------------------------------------------
Private Sub GetFormTuples()
    Dim sList As String
    Dim i As Integer, j As Integer, k As Integer
    Dim l As Integer, m As Integer, n As Integer
    Dim s As Long
    Dim buf As String
    Dim extName As String
    Dim extFile As Integer
    Dim extlen As Long
                                              ' Index in array
    n = 0

    '
    ' Do the easy one first: [Form Literal]
    '
    sList = GetProfile("Form Literal", "")    ' Get key list
    l = Len(sList)                            ' Length incl. trailing null
    i = 1                                     ' Start at 1st character
    Do While ((i < l) And (n < MAX FORM TUPLES)) ' Safety stop here
        j = InStr(i, sList, Chr$(0))          ' J -> next null
        CGI FormTuples(n).key = Mid$(sList, i, j - i) ' Get Key, then value
        CGI_FormTuples(n).value = GetProfile("Form Literal", CGI_FormTuples(n).
ey)                                           ' Bump pointer
        i = j + 1                             ' Bump array index
        n = n + 1
    Loop
    '
    ' Now do the external ones: [Form External]
    '
    sList = GetProfile("Form External", "")   ' Get key list
    l = Len(sList)                            ' Length incl. trailing null
    i = 1                                     ' Start at 1st character
    extFile = FreeFile
    Do While ((i < l) And (n < MAX FORM TUPLES)) ' Safety stop here
        j = InStr(i, sList, Chr$(0))          ' J -> next null
        CGI FormTuples(n).key = Mid$(sList, i, j - i) ' Get Key, then pathname
        buf = GetProfile("Form External", CGI_FormTuples(n).key)
```

```
            k = InStr(buf, " ")                     ' Split file & length
            extName = Mid$(buf, 1, k - 1)              ' Pathname
            k = k + 1
            extlen = CLng(Mid$(buf, k, Len(buf) - k + 1)) ' Length
            '
            ' Use feature of GET to read content in one call
            '
            Open extName For Binary Access Read As #extFile
            CGI FormTuples(n).value = String$(extlen, " ") ' Breathe in...
            Get #extFile, , CGI_FormTuples(n).value 'GULP!
            Close #extFile
            i = j + 1                            ' Bump pointer
            n = n + 1                            ' Bump array index
        Loop

        CGI NumFormTuples = n                    ' Number of fields decoded
        n = 0                                    ' Reset counter
        '
        ' Next, the [Form Huge] section. Will this ever get executed?
        '
        sList = GetProfile("Form Huge", "")      ' Get key list
        l = Len(sList)                           ' Length incl. trailing null
        i = 1                                    ' Start at 1st character
        Do While ((i < l) And (n < MAX FORM TUPLES)) ' Safety stop here
            j = InStr(i, sList, Chr$(0))         ' J -> next null
            CGI HugeTuples(n).key = Mid$(sList, i, j - i) ' Get Key
            buf = GetProfile("Form Huge", CGI HugeTuples(n).key) ' "offset length"
            k = InStr(buf, " ")                  ' Delimiter
            CGI HugeTuples(n).offset = CLng(Mid$(buf, 1, (k - 1)))
            CGI HugeTuples(n).length = CLng(Mid$(buf, k, (Len(buf) - k + 1)))
            i = j + 1                            ' Bump pointer
            n = n + 1                            ' Bump array index
        Loop

        CGI_NumHugeTuples = n                    ' Fill in global count

        n = 0                                    ' Reset counter
        '
        ' Finally, the [Form File] section.
        '
        sList = GetProfile("Form File", "")      ' Get key list
        l = Len(sList)                           ' Length incl. trailing null
        i = 1                                    ' Start at 1st character
        Do While ((i < l) And (n < MAX FILE TUPLES)) ' Safety stop here
            j = InStr(i, sList, Chr$(0))         ' J -> next null
            CGI FileTuples(n).key = Mid$(sList, i, j - i) ' Get Key
            buf = GetProfile("Form File", CGI FileTuples(n).key)
            ParseFileValue buf, CGI FileTuples(n)  ' Complicated, use Sub
            i = j + 1                            ' Bump pointer
            n = n + 1                            ' Bump array index
        Loop

        CGI_NumFileTuples = n                    ' Fill in global count

    End Sub

    '---------------------------------------------------------------------
    '
    '   GetProfile() - Get a value or enumerate keys in CGI_Profile file
    '
    ' Get a value given the section and key, or enumerate keys given the
    ' section name and "" for the key. If enumerating, the list of keys for
    ' the given section is returned as a null-separated string, with a
    ' double null at the end.
    '
    ' VB handles this with flair! I couldn't believe my eyes when I tried this.
    '---------------------------------------------------------------------
    Private Function GetProfile(sSection As String, sKey As String) As String
    Dim retLen As Long
    Dim buf As String * ENUM_BUF_SIZE

    If sKey <> "" Then
        retLen = GetPrivateProfileString(sSection, sKey, "", buf, ENUM_BUF_SIZE,
 CGI ProfileFile)
    Else
        retLen = GetPrivateProfileString(sSection, 0&, "", buf, ENUM_BUF_SIZE, C
GI ProfileFile)
    End If
    If retLen = 0 Then
        GetProfile = ""
    Else
        GetProfile = Left$(buf, retLen)
    End If

End Function
'---------------------------------------------------------------------
'
' Get the value of a "small" form field given the key
'
' Signals an error if field does not exist
'
'---------------------------------------------------------------------
Function GetSmallField(key As String) As String
    Dim i As Integer

    For i = 0 To (CGI NumFormTuples - 1)
        If CGI FormTuples(i).key = key Then
            GetSmallField = Trim$(CGI FormTuples(i).value)
            Exit Function           ' ** DONE **
        End If
    Next i
    '
    ' Field does not exist
    '
    Error ERR_NO_FIELD
End Function
'-----------------------------------------------------------------------------
'
'   InitializeCGI() - Fill in all of the CGI variables, etc.
'
' Read the profile file name from the command line, then fill in
' the CGI globals, the Accept type list and the Extra headers list.
' Then open the input and output files.
'
' Returns True if OK, False if some sort of error. See ReturnError()
' for info on how errors are handled.
'
' NOTE: Assumes that the CGI error handler has been armed with On Error
'-----------------------------------------------------------------------------
Sub InitializeCGI()
    Dim sect As String
    Dim argc As Integer
    Static argv(MAX CMDARGS) As String
    Dim buf As String

    CGI_DebugMode = True    ' Initialization errors are very bad

    '
    ' Parse the command line. We need the profile file name (duh!)
    ' and the output file name NOW, so we can return any errors we
    ' trap. The error handler writes to the output file.
    '
    argc = GetArgs(argv())
    CGI ProfileFile = argv(0)
    sect = "CGI"
    CGI ServerSoftware = GetProfile(sect, "Server Software")
    CGI ServerName = GetProfile(sect, "Server Name")
    CGI RequestProtocol = GetProfile(sect, "Request Protocol")
    CGI ServerAdmin = GetProfile(sect, "Server Admin")
    CGI Version = GetProfile(sect, "CGI Version")
    CGI RequestMethod = GetProfile(sect, "Request Method")
    buf = GetProfile(sect, "Request Keep-Alive")      ' Y or N
    If (Left$(buf, 1) = "Y") Then                     ' Must start with Y
        CGI_RequestKeepAlive = True
    Else
        CGI_RequestKeepAlive = False
    End If
    CGI LogicalPath = GetProfile(sect, "Logical Path")
    CGI PhysicalPath = GetProfile(sect, "Physical Path")
    CGI ExecutablePath = GetProfile(sect, "Executable Path")
    CGI QueryString = GetProfile(sect, "Query String")
    CGI RemoteHost = GetProfile(sect, "Remote Host")
    CGI RemoteAddr = GetProfile(sect, "Remote Address")
    CGI RequestRange = GetProfile(sect, "Request Range")
    CGI Referer = GetProfile(sect, "Referer")
    CGI From = GetProfile(sect, "From")
    CGI UserAgent = GetProfile(sect, "User Agent")
    CGI AuthUser = GetProfile(sect, "Authenticated Username")
    CGI AuthPass = GetProfile(sect, "Authenticated Password")
    CGI AuthRealm = GetProfile(sect, "Authentication Realm")
    CGI AuthType = GetProfile(sect, "Authentication Method")
    CGI ContentType = GetProfile(sect, "Content Type")
    buf = GetProfile(sect, "Content Length")
    If buf = "" Then
        CGI_ContentLength = 0
    Else
        CGI_ContentLength = CLng(buf)
    End If
    buf = GetProfile(sect, "Server Port")
    If buf = "" Then
        CGI_ServerPort = -1
    Else
        CGI_ServerPort = CInt(buf)
    End If

    sect = "System"
    CGI ContentFile = GetProfile(sect, "Content File")
    CGI OutputFile = GetProfile(sect, "Output File")
    CGI OutputFN = FreeFile
    Open CGI OutputFile For Output Access Write As #CGI_OutputFN
    buf = GetProfile(sect, "GMT Offset")
    If buf <> "" Then                                ' Protect against errors
        CGI_GMTOffset = CVDate(Val(buf) / 86400#)    ' Timeserial GMT offset
    Else
        CGI_GMTOffset = 0
    End If
    buf = GetProfile(sect, "Debug Mode")     ' Y or N
    If (Left$(buf, 1) = "Y") Then            ' Must start with Y
        CGI_DebugMode = True
    Else
        CGI_DebugMode = False
    End If

    GetAcceptTypes            ' Enumerate Accept: types into tuples
    GetExtraHeaders           ' Enumerate extra headers into tuples
    GetFormTuples             ' Decode any POST form input into tuples

End Sub
'-----------------------------------------------------------------------
```

```
'   main() - CGI script back-end main procedure
'
' This is the main() for the VB back end. Note carefully how the error
' handling is set up, and how program cleanup is done. If no command
' line args are present, call Inter Main() and exit.
'------------------------------------------------------------------------
Sub Main()
    On Error GoTo ErrorHandler

    If Trim$(Command$) = "" Then ' Interactive start
        'MsgBox "Here"
        Inter Main                  ' Call interactive main
        Exit Sub                    ' Exit the program
    End If

    InitializeCGI        ' Create the CGI environment

    '===========
    CGI Main             ' Execute the actual "script"
    '===========

Cleanup:
    Close #CGI OutputFN
    Exit Sub                        ' End the program
'------------
ErrorHandler:
    Select Case Err                 ' Decode our "user defined" errors
        Case ERR NO FIELD:
            ErrorString = "Unknown form field"
        Case Else:
            ErrorString = Error$    ' Must be VB error
    End Select

    ErrorString = ErrorString & " (error #" & Err & ")"
    On Error GoTo 0                 ' Prevent recursion
    ErrorHandler (Err)              ' Generate HTTP error result
    Resume Cleanup
'------------
End Sub

'------------------------------------------------------------------------
'
' Send() - Shortcut for writing to output file
'
'------------------------------------------------------------------------
Sub Send(s As String)
    Print #CGI_OutputFN, s
End Sub

'------------------------------------------------------------------------------
'
'   SendNoOp() - Tell browser to do nothing.
'
' Most browsers will do nothing. Netscape 1.0N leaves hourglass
' cursor until the mouse is waved around. Enhanced Mosaic 2.0
' oputs up an alert saying "URL leads nowhere". Your results may
' vary...
'
'------------------------------------------------------------------------------
Sub SendNoOp()

    Send ("HTTP/1.0 204 No Response")
    Send ("Server: " + CGI_ServerSoftware)
    Send ("")

End Sub

'------------------------------------------------------------------------------
```

```
'
'   WebDate - Return an HTTP/1.0 compliant date/time string
'
' Inputs:   t = Local time as VB Variant (e.g., returned by Now())
' Returns:  Properly formatted HTTP/1.0 date/time in GMT
'---------------------------------------------------------------------------
Function WebDate(dt As Variant) As String
    Dim t As Variant

    t = CVDate(dt - CGI GMTOffset)          ' Convert time to GMT
    WebDate = Format$(t, "ddd dd mmm yyyy hh:mm:ss") & " GMT"

End Function



'---------------------------------------------------------------------------
'
' PlusToSpace() - Remove plus-delimiters from HTTP-encoded string
'
'---------------------------------------------------------------------------
Public Sub PlusToSpace(s As String)
    Dim i As Integer

    i = 1
    Do While True
        i = InStr(i, s, "+")
        If i = 0 Then Exit Do
        Mid$(s, i) = " "
    Loop

End Sub
'---------------------------------------------------------------------------
'
' Unescape() - Convert HTTP-escaped string to normal form
'
'---------------------------------------------------------------------------
Public Function Unescape(s As String)
    Dim i As Integer, l As Integer
    Dim c As String

    If InStr(s, "%") = 0 Then                   ' Catch simple case
        Unescape = s
        Exit Function
    End If

    l = Len(s)
    Unescape = ""
    For i = 1 To l                              ' Next character
        c = Mid$(s, i, 1)
        If c = "%" Then
            If Mid$(s, i + 1, 1) = "%" Then
                c = "%"
                i = i + 1                       ' Loop increments too
            Else
                c = x2c(Mid$(s, i + 1, 2))      ' Loop increments too
                i = i + 2
            End If
        End If
        Unescape = Unescape & c
    Next i

End Function
'---------------------------------------------------------------------------
```

```
' x2c() - Convert hex-escaped character to ASCII
'
'---------------------------------------------------------------------------
Private Function x2c(s As String) As String
    Dim t As String

    t = "&H" & s
    x2c = Chr$(CInt(t))

End Function

Private Sub ParseFileValue(buf As String, ByRef t As FileTuple)
    Dim i, j, k, l As Integer

    l = Len(buf)

    i = InStr(buf, " ")                          ' First delimiter
    t.file = Mid$(buf, 1, (i - 1))               ' [file]
    t.file = Mid$(t.file, 2, Len(t.file) - 2)     ' file

    j = InStr((i + 1), buf, " ")                 ' Next delimiter
    t.length = CLng(Mid$(buf, (i + 1), (j - i - 1)))
    i = j

    j = InStr((i + 1), buf, " ")                 ' Next delimiter
    t.type = Mid$(buf, (i + 1), (j - i - 1))
    i = j

    j = InStr((i + 1), buf, " ")                 ' Next delimiter
    t.encoding = Mid$(buf, (i + 1), (j - i - 1))
    i = j

    t.name = Mid$(buf, (i + 1), (l - i - 1))     ' [name]
    t.name = Mid$(t.name, 2, Len(t.name) - 1)    ' name

End Sub
'-----------------------------------------------------------------------------
'
'   FindExtraHeader() - Get the text from an "extra" header
'
' Given the extra header's name, return the stuff after the ":"
' or an empty string if not there.
'-----------------------------------------------------------------------------
Public Function FindExtraHeader(key As String) As String
    Dim i As Integer

    For i = 0 To (CGI NumExtraHeaders - 1)
        If CGI ExtraHeaders(i).key = key Then
            FindExtraHeader = Trim$(CGI ExtraHeaders(i).value)
            Exit Function                   ' ** DONE **
        End If
    Next i
    '
    ' Not present, return empty string
    '
    FindExtraHeader = ""
End Function
```

```
Option Explicit
Global Const SystemTitle = "Invent 1.0"
Dim sSelector As String
Dim db As Database
Dim qd As QueryDef
Dim ds As Dynaset
Dim FCCRequired As String, FCCConditions As String, FDARequired As String


Function EnumerateQueryDef() As Integer
    Dim MyQuery As QueryDef
    Dim i As Integer
    Set MyQuery = db.CreateQueryDef("This is a test")
    Debug.Print
    ' Enumerate QueryDef objects.
    Debug.Print
    For i = 0 To db.QueryDefs.Count - 1
        Debug.Print Str(i) & " >" & db.QueryDefs(i).name
    Next i
    Debug.Print
    ' Enumerate built-in properties of MyQuery.
    Debug.Print "MyQuery.Name: "; MyQuery.name
    Debug.Print "MyQuery.DateCreated: "; MyQuery.DateCreated
    Debug.Print "MyQuery.LastUpdated: "; MyQuery.LastUpdated

Debug.Print "MyQuery.SQL: "; MyQuery.SQL
    Debug.Print "MyQuery.ODBCTimeout: "; MyQuery.ODBCTimeout
    Debug.Print "MyQuery.Updatable: "; MyQuery.Updatable
    Debug.Print "MyQuery.Type: "; MyQuery.type
    Debug.Print "MyQuery.Connect: "; MyQuery.Connect
    Debug.Print "MyQuery.ReturnsRecords: "; MyQuery.ReturnsRecords
    db.QueryDefs.Delete "This is a test"
    EnumerateQueryDef = True
End Function






Sub CGI Main()
    Dim X As Integer
    sSelector = UCase$(Mid$(CGI_LogicalPath, 2))    ' Remove leading "/"

    Set db = OpenDatabase("c:\website\cgi-win\db1.mdb")
    Send ("Content-type: text/html")
    Send ("X-CGI-prog: NCR Secure HTML")
    Send ("<Body>")
    Send ("")

    Select Case UCase$(CGI_RequestMethod)
        Case "GET":
            DoGet
        Case "POST":
            DoPost
        Case Else:
            Send ("<H2>Cannot do """ & CGI_RequestMethod & """ method</H2>")
    End Select
    Send ("</Body>")
    db.Close

End Sub

' Ask yourself:
```

```
'  Why did I use CGI ExecutablePath?
'  Could I have used SnapShots here?
'
Sub DoGet()
    Dim LinkStart As String
    Dim PreResults As String, PostResults As String
    Dim Results As Snapshot, i As Integer
    LinkStart = "<A HREF=""" & CGI_ExecutablePath
    Select Case sSelector
        Case ""
                 'get defined text from database
                 Set Results = db.CreateSnapshot("select distinct [file name] fro
m files")
                 Send ("<BODY>")
                 Send ("<Form method=post action=/cgi-win/invent.exe/getfile>")
                 Results.MoveLast
                 Results.MoveFirst
                 Send ("Select a file name from the list<br>")
                 Send ("<SELECT size=5 NAME=""origin"">")
                 For i = 0 To Results.RecordCount - 1
                     If Results![file name] <> "index.htm" Then
                         Send ("<OPTION>" & Results![file name])
                     End If
                     Results.MoveNext
                 Next i
                 Send ("</Select><br>")
                 Results.Close
                 Send ("You must enter a username and password to get access to t
hese files.<br>")
                 Send ("<pre>")
                 Send ("User name:  <input type=text name=username><br>")
                 Send ("Password:   <input type=password name=password><br>")
                 Send ("<INPUT TYPE=SUBMIT VALUE=""Get File"" NAME=""submit"">")
                 Send ("</PRE></FORM><br>")
                 Send ("</BODY>")
        Case Else:
            Send ("<H2>Bad GET selector """ & sSelector & """</H2>")

    End Select

End Sub

'
' Notes:
'   The real challenge is error handling. Only the simplest is done here.
'   The database is defined to prevent duplicate student & class names
'   The database is defined to enforce relational integrity
'
Sub DoPost()
    Dim X As Integer
    Dim a As Integer, okerror As Integer
    Dim i As Integer
    Dim Results As Snapshot
    Dim FSecurity As Integer
    Dim usersecurity As String, myusergroup As String
    Dim username As String, password As String
    Dim filename As String, FileSecurity As String
    Dim fileusergroup As String, temp As String
    Dim GroupSecurity As Integer
    Dim MyUserGroups() As String
    Dim FileUserGroups() As String
    On Error GoTo OnPostError       ' We need to handle errors here

    ReDim MyUserGroups(100)
    ReDim FileUserGroups(100)
    FSecurity = False

        Select Case sSelector
        Case "GETFILE"
                'get username and password
                username = GetSmallField("username")
                password = GetSmallField("password")

                Set Results = db.CreateSnapshot("select * from users where [user
 id] = '" & username & "' and password = '" & password & "'")
                If Results.EOF Then
                    Send ("<body>")
                    Send ("<h1>User Name and Password Invalid</h1>")
                    Send ("</body>")
                Else

                usersecurity = UCase(Results!security)
                myusergroup = Results![User Group]

                Results.Close
                'MsgBox myusergroup

                'get filename
                filename = GetSmallField("origin")

                'get filename security
                Set Results = db.CreateSnapshot("select * from files where [File
 Name] = '" & filename & "'")
                FileSecurity = UCase(Results!security)
                fileusergroup = Results![User Group]

                'check usersecurity against filesecurity and if it is ok then co
ntinue.
                If (usersecurity = "HI" And (FileSecurity = "HI" Or FileSecurity
 = "MEDIUM" Or FileSecurity = "LO")) Then
                    FSecurity = True
                Else
                    If (usersecurity = "MEDIUM" And (FileSecurity = "MEDIUM" Or
FileSecurity = "LO")) Then
                        FSecurity = True
                    Else
                        If (usersecurity = "LO" And FileSecurity = "LO") Then
                            FSecurity = True
                        End If
                    End If
                End If

                If FSecurity = False Then
                    Send ("<body>")
                    Send ("You do not have the correct file security<br>")
                    Send ("</body>")
                Else

                '----------
                'get group security for both the user and the file selected
                '----------
                 'fill in myusergroup array
                    a = 0
                    For i = 1 To Len(myusergroup)
                        If Mid(myusergroup, i, 1) = "," Then
                            MyUserGroups(a) = temp
                            a = a + 1
                            temp = ""
                        Else
                            temp = temp & Mid(myusergroup, i, 1)
                        End If
                    Next i
                    'get last one
                    MyUserGroups(a) = temp
                    'fill in fileusergroup array
                    temp = ""
                    a = 0
                    For i = 1 To Len(fileusergroup)
                        If Mid(fileusergroup, i, 1) = "," Then
                            FileUserGroups(a) = temp
                            a = a + 1
                            temp = ""
                        Else
                            temp = temp & Mid(fileusergroup, i, 1)
                        End If
                    Next i
                    'get last one
                    FileUserGroups(a) = temp

                '-------------
                'check group permissions, remember you are using arrays here
                '-------------

                    For i = 0 To 100
                        If MyUserGroups(i) <> "" Then

                            For a = 0 To 100

                                If FileUserGroups(a) <> "" Then

                                    If Val(MyUserGroups(i)) = Val(FileUserGroups
(a)) Then
                                        GroupSecurity = True
                                        'msgbox "groupsecurity is true"
                                        Exit For
                                    Else
                                        a = a + 1
                                    End If
                                Else
                                    Exit For
                                End If
                            Next a

                            If GroupSecurity = True Then
                                Exit For
                            End If
                        Else
                            Exit For
                        End If
                    Next i

                    '--------
                    'done checking arrays
                    'send results if true send html for the file else get out wi
th error
                    '--------
                    If GroupSecurity = True Then
                        Send (Results!html)
                    Else
                        Send ("<body>")
                        Send ("You do not belong to the correct Group, Sorry<br>
")
                        Send ("</body>")
                    End If
                    Results.Close
                End If
                Send ("")
                End If
        Case Else:
            Send ("<H2>Bad POST selector """ & sSelector & """</H2>")
```

```
DoPostFinish:                           ' Can come here via error,
                                        ' State of ds & qd unknown
    On Error Resume Next                ' Make sure ds and qd are closed
    ds.Close                            ' else db.Close will fail and you lose
    qd.Close

    Exit Sub

' ================
' Exception Handler
' ================
'
OnPostError:
    If Err = ERR NO FIELD Then
        okerror = ERR_NO_FIELD
        Resume Next
    End If

    If Err >= CGI_ERR_START Then Error Err   ' Resignal if a CGI.BAS error

    Send ("<H2>There was a problem:</H2>")
    Send ("VB reports: <CODE>" & Error$ & " (error #" & Err & ")</CODE><H3>Best
Guess:")

    Select Case sSelector
        Case "ENROLL":         ' Probably a duplicate name (enforced by database)
            Send ("Already enrolled")

        Case "DISMISS":        ' This is ugly, name came from dropbox
            Send ("?? This is ugly ??")

        Case "ADD":
            Send ("Class already exists")

        Case "DEL":
            Send ("?? This is ugly ??")

        Case "CL4ST":
            Send ("?? This is ugly ??")

        Case "ST4CL"
            Send ("?? This is ugly ??")

        Case "TAKE":
            Send ("Already taking this class")

        Case "DROP":
            Send ("Not in this class")

        Case Else:
            Send ("Programmer error: Unknown selector in POST exception handler.
")
    End Select

    Send ("</H3>")

    Resume DoPostFinish

End Sub

Sub Inter_Main()
CGI Main
    MsgBox "This is a Windows CGI program"
```

```
Module2 - 6

Sub OptionList(FieldName As String, Tbl As String, Col As String)

    Send ("Select " & FieldName & ": <SELECT NAME=""" & FieldName & """>")
    Set ds = db.CreateDynaset(Tbl)
    Do Until ds.EOF
        Send ("<OPTION>" & ds(Col))
        ds.MoveNext
    Loop
    ds.Close
    Send ("</SELECT>")

End Sub



Public Function ConvertSpaces(temp As String)

End Function

Public Function ConvertPlusSigns(temp As String)

End Function
```

Claims

1. A document security system comprising a server (12A) in which a plurality of documents are stored for access by user terminals (20A..20N)

   characterised in that

   a database (A) is provided in the server (12A), which database (A) has: means for storing user information (110,120,130,140,150); means for storing document information (210,220,230,240); and means for providing access to the stored documents document-by-document on the basis of the user information and the document information.

2. The system according to claim 1 wherein the said means for storing user information includes means for storing a user identification name (110), an associated user password (130) and an associated security level indicator (140) for indicating the highest level of security access for the user name associated therewith.

3. The system according to either one of the preceding claims wherein the said means for storing document information includes a file name (230), code means for creating a document (240) associated with the file name (230) and a security level indicator (210) associated with the file name (230) for indicating the security level of the associated document (230).

4. The system according to any one of the preceding claims wherein the said means for providing access to stored documents is included in a common gateway interface file (CGI-A..CGI-N).

5. The system according to any one of the preceding claims and comprising a plurality of different servers (12A..12N) each having its own database (A..N) and each having an internet connection to enable any of a plurality of user terminals (20A..20N) to be connected to any of the servers (12A..12N).

6. A method of providing document security in an environment where a server stores a plurality of documents and the server is accessible by any of a plurality of user terminals comprising the steps of:

   assigning a security level to each document,

assigning a security level to each user terminal,
receiving a request at the server from a user terminal for access to a document,
determining the security level assigned to the user terminal,
comparing the determined security level with the security level assigned to the requested document, and
providing access to the requested document only if the result of the comparison step indicates that the security level of the said user terminal is at least as high as the security level assigned to the requested document.

7. The method according to claim 6 wherein there are a plurality of servers and including the step of locating the particular server in which the requested document is stored.

8. The method according to claim 6 or claim 7 and including the step of associating a user identification name and a user password with the assigned user security level.

FIG. 1

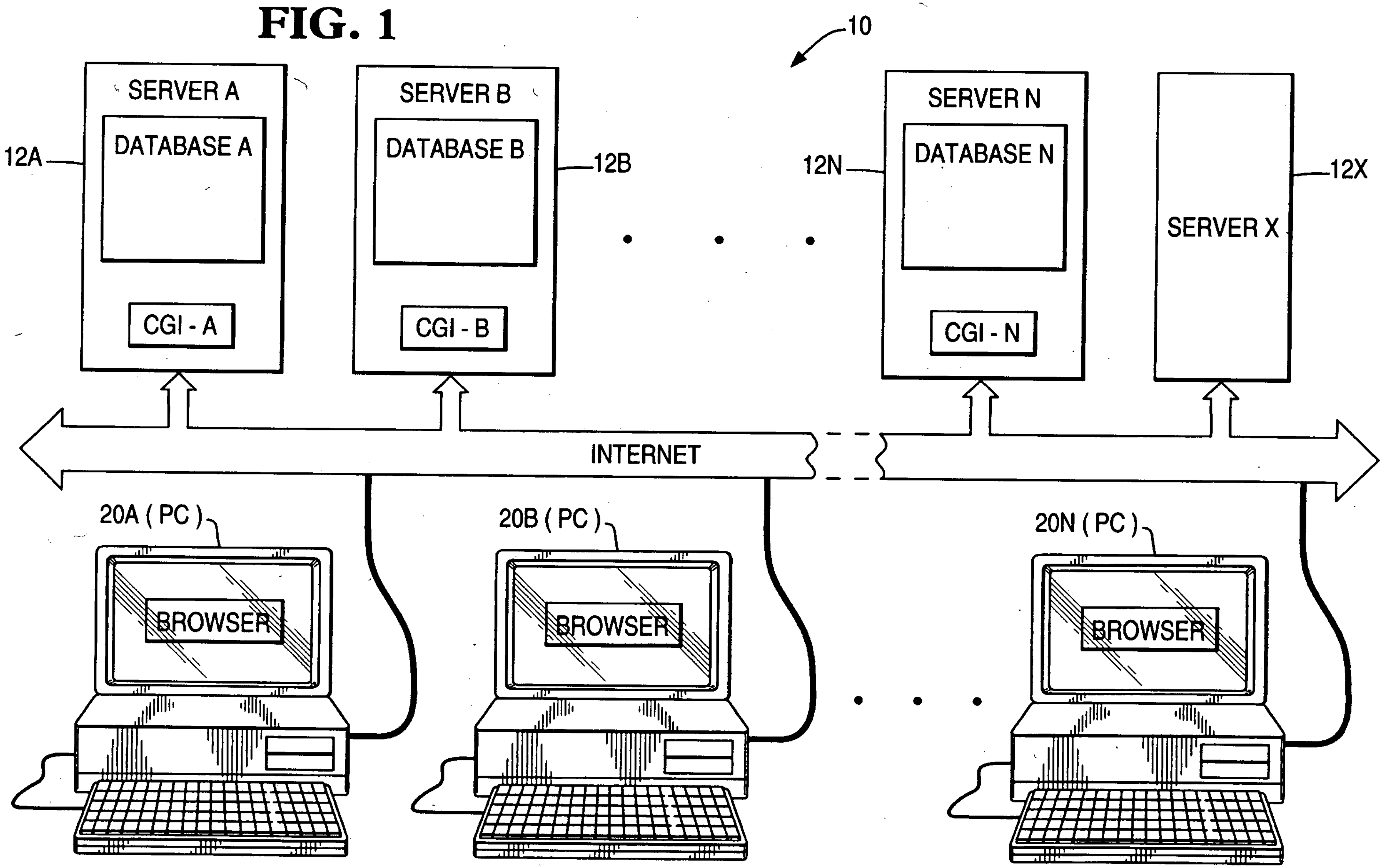
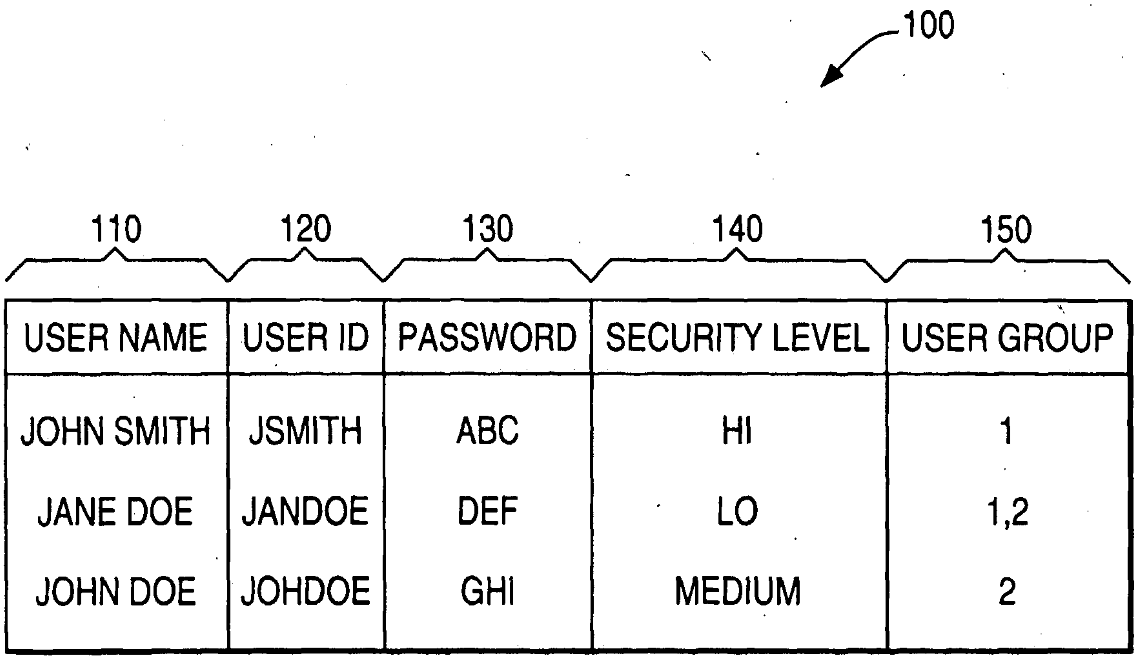

10
SERVER A
DATABASE A
CGI - A
12A
SERVER B
DATABASE B
CGI - B
12B
SERVER N
DATABASE N
CGI - N
12N
SERVER X
12X
INTERNET
20A (PC)
BROWSER
20B (PC)
BROWSER
20N (PC)
BROWSER

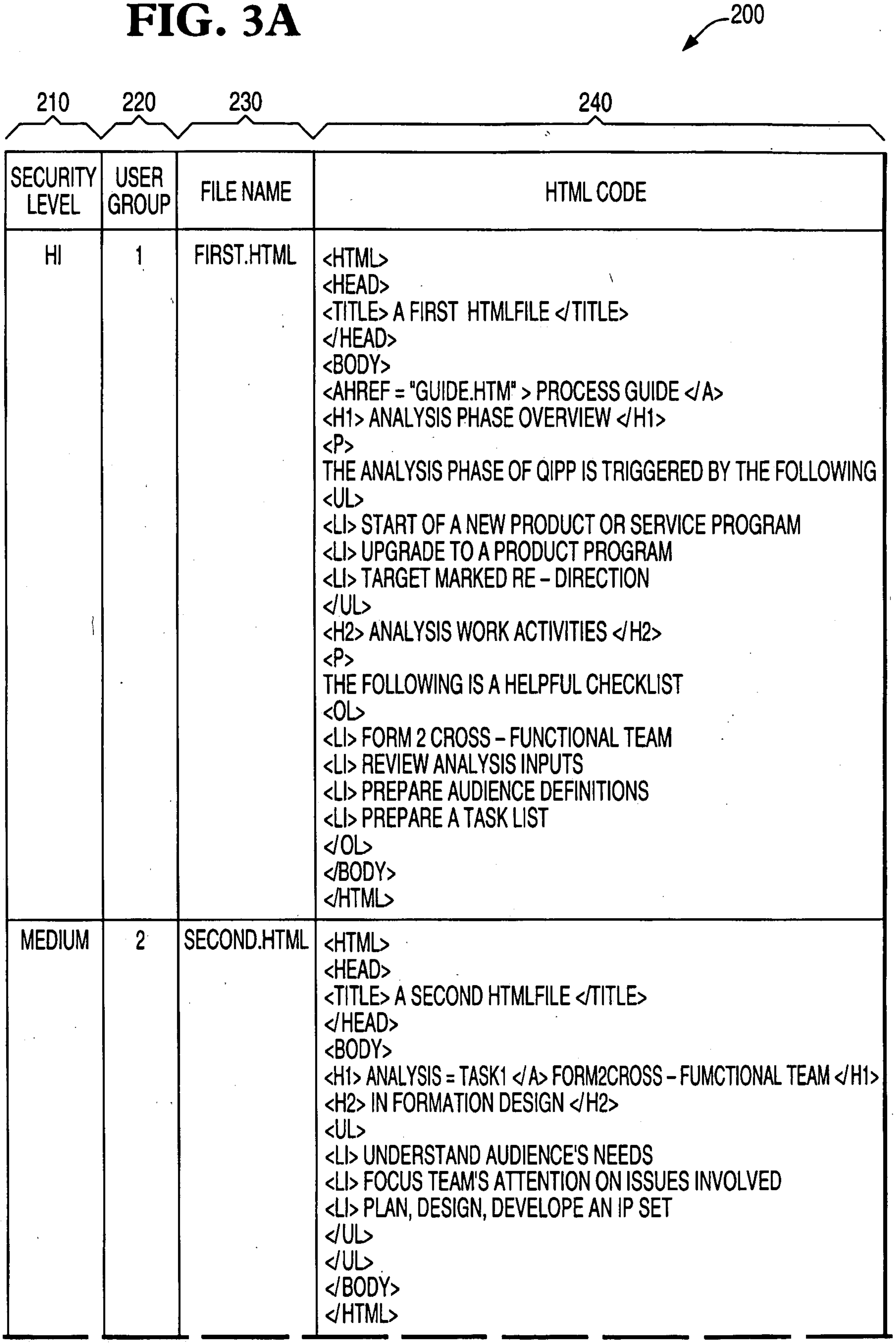
FIG. 2



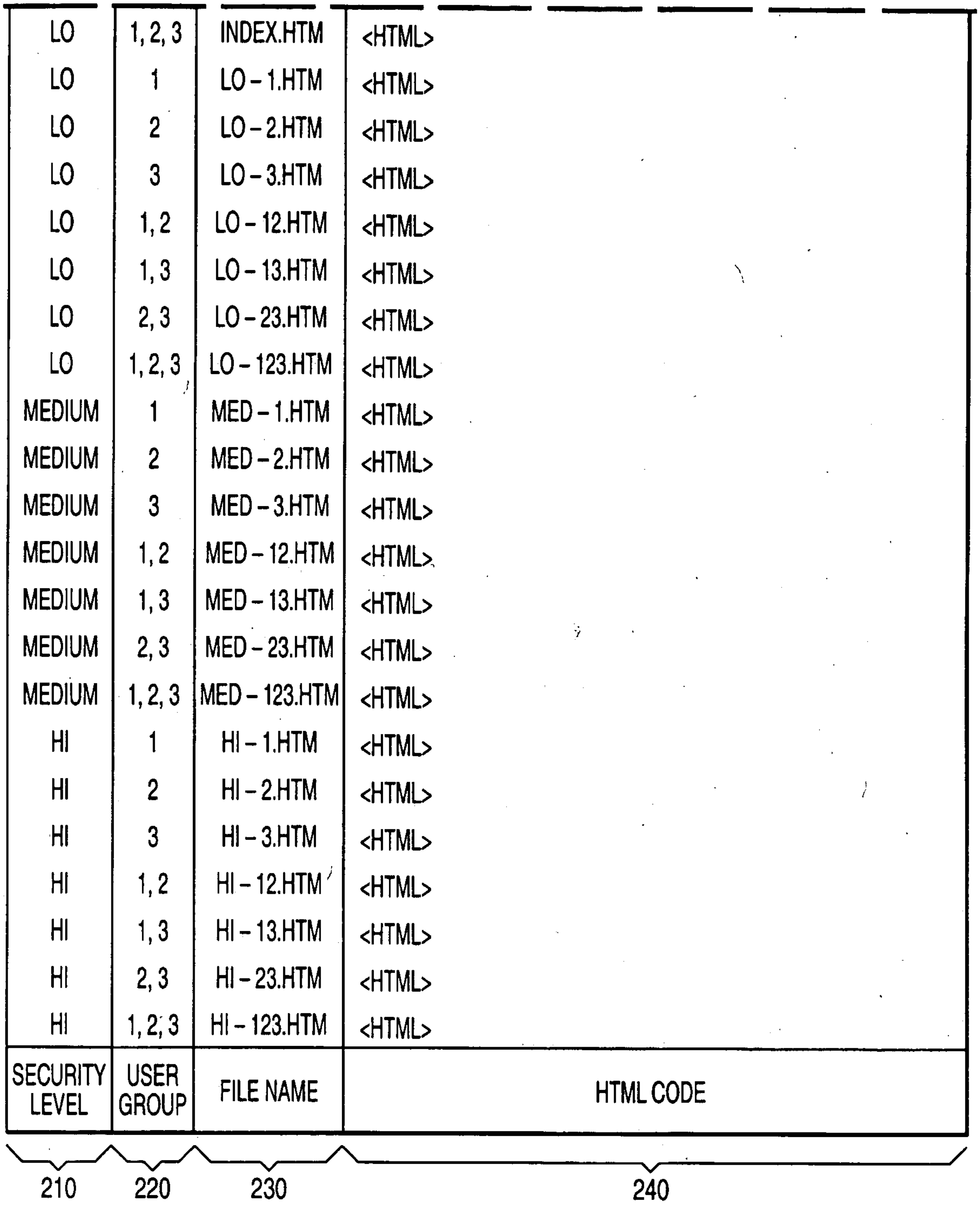
USER NAME	USER ID	PASSWORD	SECURITY LEVEL	USER GROUP
JOHN SMITH	JSMITH	ABC	HI	1
JANE DOE	JANDOE	DEF	LO	1,2
JOHN DOE	JOHDOE	GHI	MEDIUM	2

100
110
120
130
140
150

FIG. 3A


200
210
220
230
240

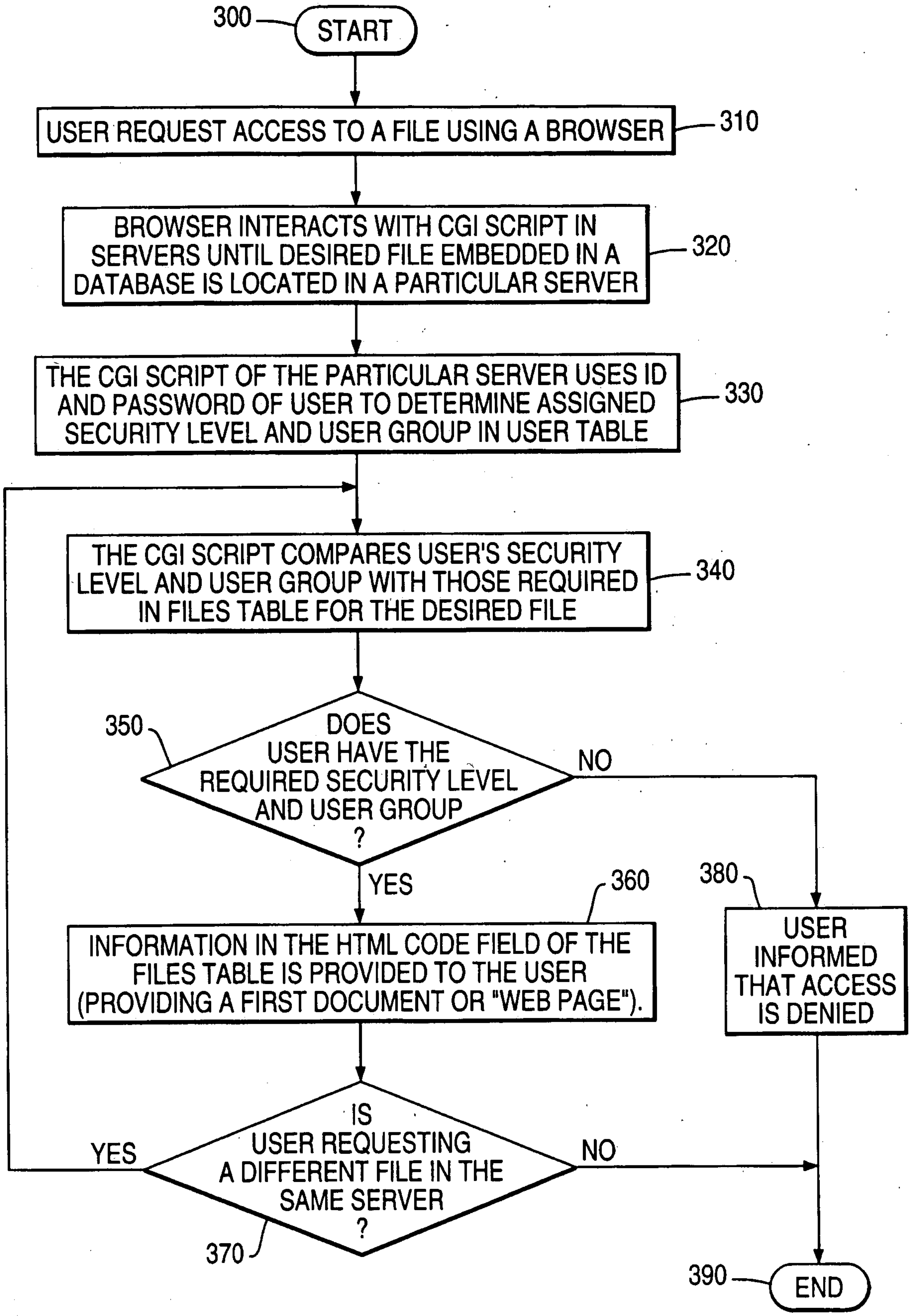
SECURITY LEVEL	USER GROUP	FILE NAME	HTML CODE
HI	1	FIRST.HTML	<HTML> <HEAD> <TITLE> A FIRST HTMLFILE </TITLE> </HEAD> <BODY> <AHREF = "GUIDE.HTM" > PROCESS GUIDE </A> <H1> ANALYSIS PHASE OVERVIEW </H1> <P> THE ANALYSIS PHASE OF QIPP IS TRIGGERED BY THE FOLLOWING <UL> <LI> START OF A NEW PRODUCT OR SERVICE PROGRAM <LI> UPGRADE TO A PRODUCT PROGRAM <LI> TARGET MARKED RE – DIRECTION </UL> <H2> ANALYSIS WORK ACTIVITIES </H2> <P> THE FOLLOWING IS A HELPFUL CHECKLIST <OL> <LI> FORM 2 CROSS – FUNCTIONAL TEAM <LI> REVIEW ANALYSIS INPUTS <LI> PREPARE AUDIENCE DEFINITIONS <LI> PREPARE A TASK LIST </OL> </BODY> </HTML>
MEDIUM	2	SECOND.HTML	<HTML> <HEAD> <TITLE> A SECOND HTMLFILE </TITLE> </HEAD> <BODY> <H1> ANALYSIS = TASK1 </A> FORM2CROSS – FUMCTIONAL TEAM </H1> <H2> IN FORMATION DESIGN </H2> <UL> <LI> UNDERSTAND AUDIENCE'S NEEDS <LI> FOCUS TEAM'S ATTENTION ON ISSUES INVOLVED <LI> PLAN, DESIGN, DEVELOPE AN IP SET </UL> </UL> </BODY> </HTML>



SECURITY LEVEL	USER GROUP	FILE NAME	HTML CODE
LO	1, 2, 3	INDEX.HTM	<HTML>
LO	1	LO – 1.HTM	<HTML>
LO	2	LO – 2.HTM	<HTML>
LO	3	LO – 3.HTM	<HTML>
LO	1, 2	LO – 12.HTM	<HTML>
LO	1, 3	LO – 13.HTM	<HTML>
LO	2, 3	LO – 23.HTM	<HTML>
LO	1, 2, 3	LO – 123.HTM	<HTML>
MEDIUM	1	MED – 1.HTM	<HTML>
MEDIUM	2	MED – 2.HTM	<HTML>
MEDIUM	3	MED – 3.HTM	<HTML>
MEDIUM	1, 2	MED – 12.HTM	<HTML>
MEDIUM	1, 3	MED – 13.HTM	<HTML>
MEDIUM	2, 3	MED – 23.HTM	<HTML>
MEDIUM	1, 2, 3	MED – 123.HTM	<HTML>
HI	1	HI – 1.HTM	<HTML>
HI	2	HI – 2.HTM	<HTML>
HI	3	HI – 3.HTM	<HTML>
HI	1, 2	HI – 12.HTM	<HTML>
HI	1, 3	HI – 13.HTM	<HTML>
HI	2, 3	HI – 23.HTM	<HTML>
HI	1, 2, 3	HI – 123.HTM	<HTML>

FIG. 3B

FIG. 4


300
START
USER REQUEST ACCESS TO A FILE USING A BROWSER
310
BROWSER INTERACTS WITH CGI SCRIPT IN SERVERS UNTIL DESIRED FILE EMBEDDED IN A DATABASE IS LOCATED IN A PARTICULAR SERVER
320
THE CGI SCRIPT OF THE PARTICULAR SERVER USES ID AND PASSWORD OF USER TO DETERMINE ASSIGNED SECURITY LEVEL AND USER GROUP IN USER TABLE
330
THE CGI SCRIPT COMPARES USER'S SECURITY LEVEL AND USER GROUP WITH THOSE REQUIRED IN FILES TABLE FOR THE DESIRED FILE
340
350
DOES USER HAVE THE REQUIRED SECURITY LEVEL AND USER GROUP ?
NO
YES
360
380
INFORMATION IN THE HTML CODE FIELD OF THE FILES TABLE IS PROVIDED TO THE USER (PROVIDING A FIRST DOCUMENT OR "WEB PAGE").
USER INFORMED THAT ACCESS IS DENIED
YES
IS USER REQUESTING A DIFFERENT FILE IN THE SAME SERVER ?
NO
370
390
END

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 30 6103

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP 0 547 990 A (IBM) * the whole document *	1	G06F1/00 G06F12/14
Y	---	2-6	
Y	YOUNG C R: "A Security Policy for a Profile-Oriented Operating System" AFIPS CONFERENCE PROCEEDINGS, 4 - 7 May 1981, CHICAGO, IL, US, pages 273-282, XP002060077 * the whole document * ---	2	
Y	WHITCROFT A ET AL: "A tangled Web of Deceit" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 2, no. 27, November 1994, page 225-234 XP004037993 * the whole document * ---	3,5,6	
Y	GODWIN-JONES R: "INTERACTIVE WEBBING: CGI SCIPTING, JAVASCRIPT AND LINKED PROGRAMS FOR LANGUAGE LEARNING" CALICO. ANNUAL SYMPOSIUM. PROCEEDINGS OF THE COMPUTER ASSISTED LANGUAGE INSTRUCTION CONSORTIUM. DISTANCE LEARNING, 27 May 1996, pages 127-131, XP000617426 * the whole document * ---	4	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F
A	US 5 291 598 A (GRUNDY GREGORY) ---		
A	US 5 319 705 A (HALTER BERNARD J ET AL) ---		
A	EP 0 398 645 A (IBM) -----		

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	24 March 1998	Powell, D

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons
& : member of the same patent family, corresponding document